Available online at www.sciencedirect.com

SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 146 (2006) 17–32

www.elsevier.com/locate/entcs

A Composite Web Service Supporting User Context to Provide an Adapted Remote Control of High Technology Instruments

Christophe GRAVIER[1]

DIOM laboratory, SATIn group
Jean Monnet University
Saint Etienne, France

Jacques FAYOLLE[2]

DIOM laboratory, SATIn group
Jean Monnet University
Saint Etienne, France

Abstract

In this paper, we are aiming at providing a complete, plate-form independent, framework, for the remote control of high technology instruments. This is lead by the idea of sharing resources (instruments in this use-case) between entities, each exploiting of course their own information system. Mainly, to be generic enough to satisfy the corresponding genericity of resources addressed, the sharing must be done under certain constraints, which are security, scalability, authentication, "real time" access, Multi-Platform and Multi-Users access. The purpose of the article is to discuss the possible use of Web services for the skeleton of such a generic framework, with the issue of providing an adapted service to the user depending on the context of utilization (i.e. depending on the role the user is playing in the session).
http://www.istase.com/satin/eINST.html.

Keywords: User-Context awareness, Security, Scalability, eLaboratories, eInstrumentation, Computer Supported Collaborative Work - CSCW, composite Web services, Message Oriented Middleware.

[1] Email: christophe.gravier@univ-st-etienne.fr
[2] Email: jacques.fayolle@univ-st-etienne.fr

doi:10.1016/j.entcs.2005.11.004

1 Introduction

The need of high technology resources for technological researches cannot be denied. It is widely accepted that many research themes *need* to get access to a *specific* high technology instrument in order to make some experiments and/or to validate their results.
Nevertheless, laboratories are usually unable to afford all the amount of instruments that they consider necessary. In addition, there may be a punctual need that will not justify an investment, but that would have been a nice observation for the theorical paradigm addressed, if the researcher have been granted a single access to a specific instrument. This laboratory context can "easily" be extended to companies issues as well as experiments / hands-on approaches for students (regarding technology teaching). What we propose is to offer a remote control mechanism of a high-tech resource, in order to share it between entities, which could be

- *Researchers* in the field of eLaboratories,
- *Companies' employees*, in the case of a partnership between company and a public university owning the instrument,
- *Students*, for use of the instrument as a support of pedagogy in specific hands-on activities.

The main point of our proposition is to provide a generic framework, independent of the local interfaces controlling the resource that would fit both research/company and teaching needs. We aim at supporting a real time communication between clients and legacy or/and native interfaces. This must be done under a strong security context due to the nature of the resources controlled. Note that security not only covers integrity but also availability and robustness. The aim of this paper is to clarify how Web services can help in the elaboration of such a framework. Constraints are numerous and are covering a large field of interest such as

- *Security* as the resource *must* not be listened, be corrupted or be used without the permission to,
- *Applications Group Awareness* for collaborative access to the instrument (Computer Supported Collaborative Work),
- *Adaptation to user* through the *context* the user is using the Web Services,
- *Scalability* since the potential instruments and users that could be involved simultaneously is important,
- *"Real-time" Asynchronous Middleware* structure. "Asynchronous" has to be understand as it is specified in the field of middleware. That means

"receive" instruction is a non-bloquant instruction, as it will be pointed out later in this article.

- *Multi-platform* since people who access the instrument are from many different origins, as listed previously (researchers, companies' employees or students).
- *Journalization* of the history of all the accesses (successful and unsuccessful) that are made to the resource.

We are focusing here on the utility and the usability of Web Services as a solution for our generic framework, and more specially on the adaptation of service to the users depending on the context (i.e. time and mean by which they are using the resource).
In order to support our purpose, we will first discuss about the design steps of our framework. This means that we will explain why and how we thought about Web services being a potential base for our solution. Then, we will focus on the user role and its adaptation regarding the context (the aim is to provide Web services adapted the user context). After those explanations, we will put all constraints we have to support towards the Web Services behavior and see if it's match the objectives. Finally, we will provide a full set of illustrations where the Web Services based framework could be employed. Future works and conclusion will end this paper.

2 Base Framework with one Web Service

Functionally speaking, the hardest function to provide may be the support of a multi-users activity. Indeed, a usually accepted way of designing Computer Supported Collaborative Work (noted CSCW from now), are included in the two major thoughts as explained in [4]

(i) *collaboration transparent*: to seem to be collaborative (the environment is shared)

(ii) *collaboration aware*: to be collaborative (the application itself is shared)

As pointed in [4], each approach has its drawbacks: a lack of a *group awareness* for the first, to be *time-consuming* for the second. Our purpose here is support collaboration using MOM [3] as described in [1], and ObjectWeb Joram[9] implementation of MOM specification more especially. This way, we are aiming at positioning a generic solution on the top of the application layer accessing the instrument. We can say that we are no longer in case i nor ii, but that we are providing an entire original approach consisting in sharing

[3] Message Oriented Middleware

an interface pointing to, not only a single application, but in fact to several applications being in the same group of interest, which is group of applications accessing an instrument at the same time (collaborative access). This interface is positioned between the environment containing the application to be accessed and the users running their own applications in their own environment. The following figure 1 can help in getting the idea developed here.

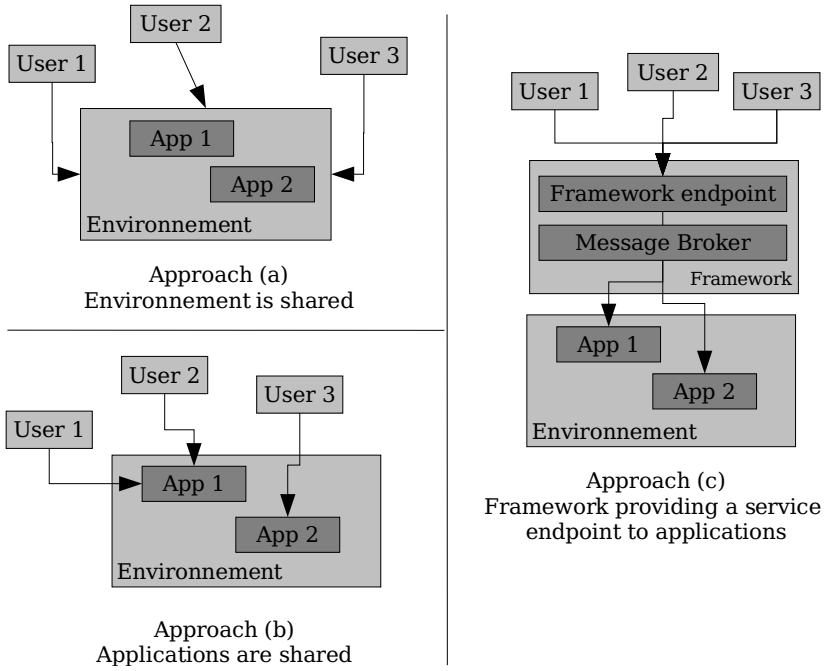

Fig. 1. Our CSCW approach: use of a framework to support collaborative accesses

We may add some words on the message broker. It is viewed here as the widespread definition: the message broker is the heart of a MOM since he is held responsible for delivering the messages to applications. When using the "publish/subscribe" mode, one creates a "topic [4] ", which is a communication channel on which applications can communicate. When a message is produced on a single topic, the message is multicasted to all applications having subscribed to channel. This is the same principle as the mailing list: if someone send a message to the list, everyone who subscribed will receive the message. In our architecture, the MOM (via its Message Broker) aims at delivering messages between the users' applications and the application controlling locally the resource (an high technology instrument in the use-case of this paper). This provides us collaborative access to the application running on the host

[4] a topic is the structure responsible for delivering the message in a "publish/subscribe" model for a Message Oriented Middleware

connected to the instrument: if both User 1 and User 2 want to access the instrument, each command will be forwarded by the MOM and the response will be multicasted to User 1 *and* User 2.

Our first idea was to use a Web service as "Framework endpoint" as described in the figure 1. Since Web Services end points are supposed to be exploited in several languages such as C or Java, and since Java (at least the Java Virtual Machine) can be used under multiple platforms, java-coded Web service with its endpoint, both providing an access to the MOM could intrinsically satisfy the multi-platform requirement. The illustration of such an idea is given on figure 2, the step of sending a command through the framework, and figure 3 illustrates the the response sent by the application through which users access to the instrument.

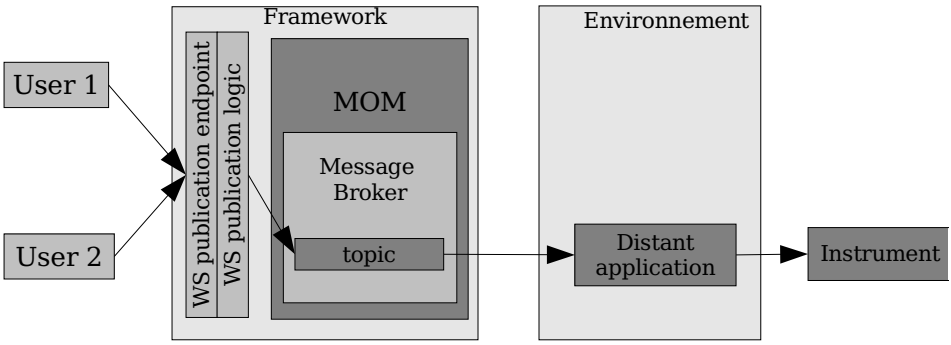

Fig. 2. Publish a command to the instrument via our framework

The figure 2 gives the steps a "command message" go through. The necessary steps are, in order they are bypassed by the messages requesting an order to the instrument (we assume that there is two users "connected" to the *eInstrument*, that means to the distant application controlling the instrument.

(i) Firstly, the user "User 1" uses his application, that virtually represents the instrument, to request a command to the distant instrument. The message is built up at this step.

(ii) User 1's application use its Web Service endpoint to address the designated Web service. Thus, the message is managed through classic SOAP serialization.

(iii) The instance of the Web service create a MOM-based message (satisfying the JMS [5] standards, we selected Java as the language for the framework).

[5] Java Messaging Service: briefly, this is the normalization which is used by all java-based

The Web service uses a specific MOM topic to publish the message.

(iv) In the end, since both User 1's plus User 2's applications, and the distant application controlling the instrument, had subscribed to the MOM topic, they are all supposed to receive the order, originally only addressed to the Instrument. In order to avoid this kind of unnecessary networks solicitations, it must be noticed that the topic structure of a MOM allows filtering (sometimes called *selectors* depending on the MOM implementation used). This way, it can be guaranteed that only the distant application will receive the message. (therefore, in the step of sending a message throw the framework, the multicasting is reduced to a point-to-point communication).

Once the message *is* delivered to the distant application, the application connected to the instrument fired two process:

- *systematic acknowledgements to the group.* This help applications accessing the same instrument, to notice their users that an order has been generated and that this command has been delivered properly to the distant application controlling the instrument. This notification is published using the Web Service. This way, Users' applications will display a visual effect suck as blinking a graphic controlling in order to notice the user that the command *had been* received.
- *treat the received message* in order to call the native method granting access to the instrument.

The acknowledgement is really a simple reflex: the application receives a message *so it fires* automatically a notification to the group.
Besides, treating the message is something more complex since it deals with the native interfaces. After that, it has to multicast the response to the applications in the group. The response is symmetrical to the sending of the command message as shown in figure 3

(i) After the operation requested is done, the distant application uses the Web service to *create the JMS message.*

(ii) The Web Service relay the result message to the MOM topic

(iii) The Message Broker hosting the topic makes a multicast for the result computed to User 1's and User 2's applications (remember the filtering operating for the topic that will make the delivery of a response only to specific sub-group of the total applications instances having subscribed it.

Message Broker. See [8]

(iv) The users' applications display the results to their respective users.

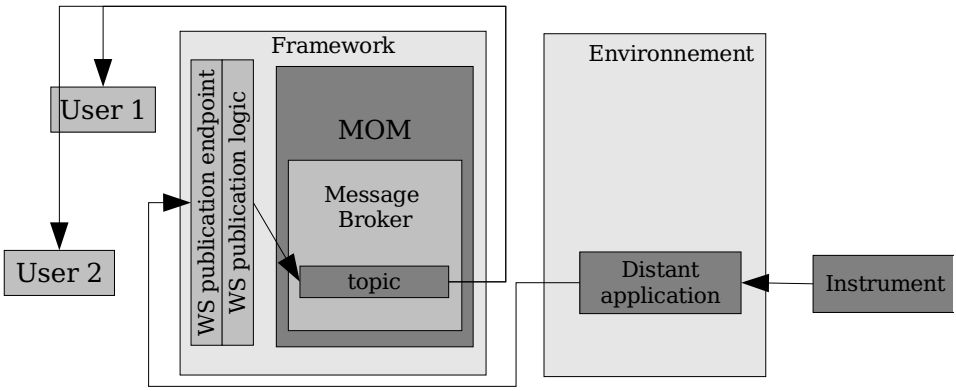

Fig. 3. Publish a response to users' applications for a request command

Summary 1 *This architecture help in building a framework for a collaborative access of an instrument. In fact, thank to it, users' are able to access the instrument with one user controlling it and others that are watching what is done by the user who is manipulating. Nevertheless, there are imperatives that could not be satisfied using this simple architecture, which are the security of the resource, scalability of the solution proposed and the logging of all the messages that are emitted.*

The next section will discuss about the ameliorations that are needed in order to satisfy those additional objectives.

3 Base Framework with composite Web Services.

Web services Collaboration mode regarding Network load

It can be deduced from the first part of this paper in section 2, that *scalability* has not been considered and is *not* satisfied by the simple architecture proposed earlier. The main purpose is that, if there is only one instance of a Web service exposing its endpoint to the framework, there is no denying that it will not be enough to support data flow, even at a low rate. We can take it for granted that if the Web service takes a burst of demands (asking for publication of a message on the MOM), it has to treat the entire process as describe in 4 before proceeding with another request. This use of a Web service is called *Collaborative Web Service Access*. In our architecture, this is something to avoid since, if there is multiple clients accessing a single instance of a web service to obtain a consistent service, the fact that there is only *one* instance offering the service cannot be enough to support a real use of the

framework. Nevertheless, this has the advantage to multicast the output of a single Web Service to multiple clients, this means playing quite the same role as a topic structure (this even have the possibility to manage late joiners).

The second approach to consider is *Collaborative Replicated Web services*, see [3]. This offers the opportunity to have replication of the Web Service's instance. This allows to support more requests by client applications. However, there is a big need of maintaining the copies consistent to synchronize all the outputs of the different instances. This should be keep for period of heavy traffic, since replication is not simple, consume resources and some communication channel time to provide synchronization.

Perhaps one could imagine to take the best of the two worlds by switching the access of a Web service from collaborative access to replicated access, depending on the context, i.e. the load of the Web service.

Nevertheless, this is already a theoretical issue: if load conditions alternate a lot near the "switch point" (i.e. the point of network where it is considered better to switch from one collaborative access to another), you need to put some kind of watchdog which will not allow to re-switch if you have switched lately. The problem of choosing the right time-limit within which the collaboration mode cannot be changed, implies a consensus problem (see [2]), that can be considered as hard one to deal with. In addition, while observing the practical implementation of such a switching process, one have to provide processes to

- make replication of a Web service's instance and initialize the corresponding event service to keep those instances synchronized while switching from approach with one instance to the approach with replicated instances.
- determine between several instances the one that is consistent, if there differences between instances, and killed the others simultaneously in order to keep only one instance.

To summarize, we can understand that trying to take Network load into account for the access of a single service point out a lot of problems and may not enlight enough responses to realize it easily. This will be quoted as future works under the corresponding section of this paper.

Composite Web Service

Apart from scalability, the architecture should aim at providing authentication and the logging of all the accesses. Basically speaking, this can be

achieved by using a directory and a corresponding protocol such as LDAP [6] for the authentication, and a DBMS [7] such as postgreSQL. It is very important to note that authentication and logging are two additional services that can (must!) be implemented as Web services. In addition, if the newly considered architecture contains three Web services, which are

- a publication Web service for publishing messages on the MOM topic,
- an authentication Web service for checking access rights on the "eInstrument" for the corresponding user,
- a logging Web service that journalizes messages vehiculed through the framework,

we can now dress a new view of our solution as shown in figure 4, which compose Web services to response to users' requests, that else could not have been satisfied by any available service. This is directly based on the observation settled in [6] since we are aiming too at providing a "combination of several independent Web Services into a single, consistent service". The combination can help too in thinking about load balancing with providing more instances of Web services for the more solicited service (probably logging which can be a longer task to execute). The composition can be realised by using BPEL4WS [8] in order to orchestrate the partnerships between those Web services.

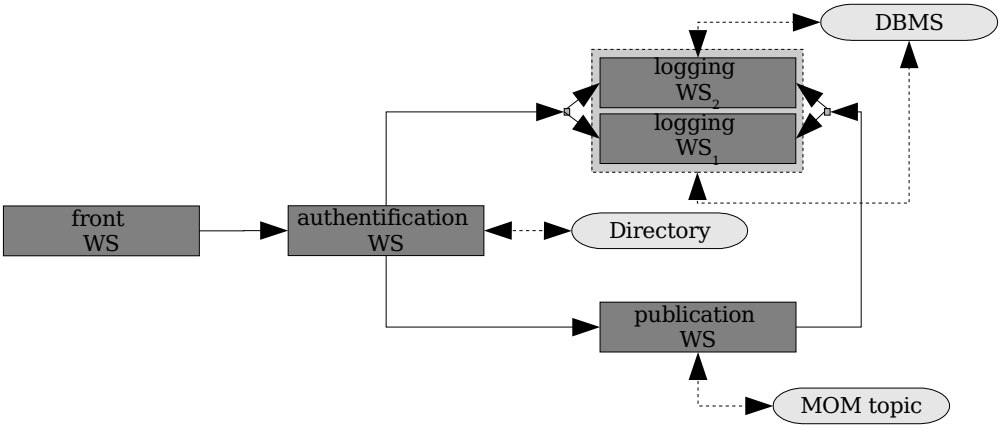

Fig. 4. a Composite Web service for the remote control of high tech instruments

Summary 2 *At first, we evaluate the solution of having replicated Web ser-*

[6] Lightweighted Directory Access Protocol
[7] DataBase Management System
[8] Business Process Execution Language for Web Services

vices instances in order to take into account the Network loads context and so offer a better "quality of service" (which means at least get adapted to the Network context) to the clients. Nevertheless, it appears to be a real challenge that could not be handle easily, just because this collaboration mode is suited for high loads, and that the adaptation (i.e. the switching between collaboration modes) is really something as difficult theoretically speaking as practically speaking. However, there was a really smart thing we encounter: the Web service providing publication on the MOM topic could not exist by itself. Indeed, it have to be surrounded by additional services which provides authentication and logging as example to meet the requirements we had exposed in 1.

4 User Context in the framework.

Determine User role from the Context.

The users will have to fire commands from a virtual representation of an instrument, which is called *eInstrument*. Regarding the context of the user, there may be different virtual representations of the instrument for different segmentation of the population using the instrument and even different representations for the same user, depending on the purpose. For example, the user plays the role a teacher for a pedagogic collaborative access to an instrument, it might be more interesting for him (and surely for students!) to only display the graphical controls that are supposed to be manipulated during the hand-ons course. On the opposite, if the same user, now acting like a researcher, want to access the instrument, it is obvious that, by default, all the functionalities have to be available for his session of manipulating. The main problem is to acquire the context, i.e. does the user is currently logging in playing the role of researcher or teacher. Several questions can be raised regarding the context:

- Does the user began a Teaching Session ?
- Does the user began a Research activity ?
- Depending on the status of the user (which can be Students, Researcher, Company's employees), does he have the same graphical control than another categories ?
- Does a company maintain a more important partnership with the entity owning the instrument, so that the amount of controls displayed will not be the same ?

Unlike other more context independant propositions (see [5]), we want here to be able to answer tose previous questions. The answers must be *automated* in order to provide a dynamic adaptation of the virtual representation of the

instrument depending on the context.
The main point is to identify the context property that can help in evaluating the context from which the user is accessing the instrument. Going back to the example of a researcher with some teaching activities. We could possibly determine from the *IP address* used by the user's application if he is accessing the instrument from the laboratory or from the experiment classroom.
One can argue that the IP address cannot fully determine that the user is accessing the instrument while endorsing the role of a teacher or of a researcher. We think that it is completly true and that the determination cannot be reduced by the evaluation of a single context property. Following the previous example, we can add that the role the user plays in the manipulation can be deduced from *the instrument he is accessing*. For our researcher/teacher user, we can easily imagine that the research activities of such a user lead him to use a eInstrument noted A, and that the user does not necessary use the same eInstrument in his practical courses with the students. In the best case, he may only access a second eInstrument noted B, in context of pedagogy, and A for research activities. This would help us in displaying the right eInstrument, that mean the right amout of graphic controls, for the user depending on the role he is playing , deduced from the context.
Nevertheless, in worst cases, we can assume that context property one (IP address) or second (the instrument being accessed), are not enough determinism to rightly evaluate the role of the user. That is the reason why we could possibly provide another context property, that can be the *time table* of the researcher/teacher. Indeed, if a Web service is able to deliver to our framework that the user is connecting on a time range when he is supposed to be in an experiment classroom, teaching students, that would most probably means that the researcher/teacher is acting as a teacher for this session (so the display of the eInstrument would be adapted).
However, there are situations, that means there are contexts, where the decision of the role being played cannot be reduced to a single context property (as discussed earlier). In addition, there could be evaluation where some context properties are more revealing than other. In this case, a ponderation mechanism on the context constraints could be proposed in order to help in choosing the right context. In our example, the time table context property may be even more revealing than the IP address which is itself more revealing than the instrument being accessed.
Our proposition, as exposed in figure 5, uses a front "context aware" Web service collaborative which decide the role of the user. To take the decision, this Web service use a composite service being built with every Web service that delivers the context property it is accessing. The ponderation mechanism can be implemented inside the front Web service in order to give more credit

to a Web service than another if the context property it delivers is more revealing than the other Web service's one. Nevertheless, the context can be so crucial for the application that it can be justified to introduce more complex method to predict the role such as ones melting graph theory and probability. Such mathematics models exist: this is for example Hidden Markov Model or Bayesian Networks.

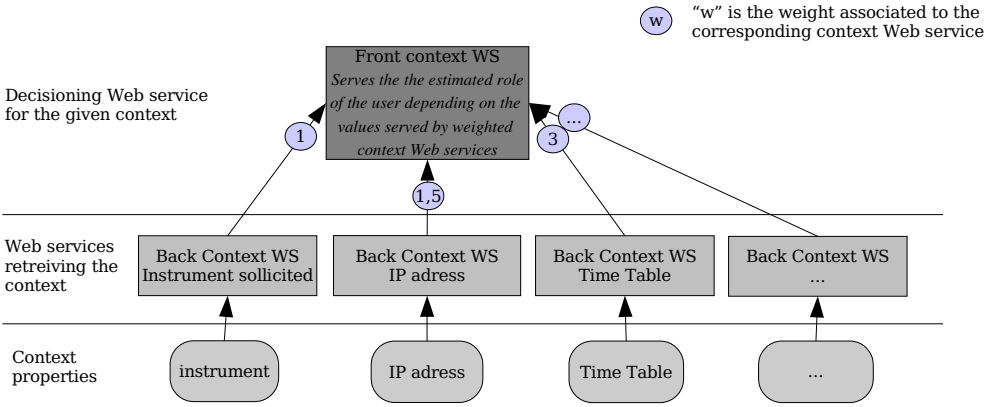

Fig. 5. The generic eInstrumentation framework supporting user-role adaptations based on context properties

In figure 5, one can easily understand the organisation by *layers* of our proposition. This illustration shows how it can be possible to implement a Web service depending on the context. In fact, the corresponding "*service-based context layer*" is viewed from an external source as a unique service but in fact it relies on a set of services retrieving and assessing the context and take a decision on the service to deliver to the user. Moreover, as said earlier, since we deal with context, there are more significant properties on which we can put the stress on, and so weight them to a corresponding value. In the example, it is clear that knowing that a researcher/teacher *is* in a course at the given period of time is a very significant clue on the role he wants to play towards a context propriety such as the instrument that is being accessed.

Integration of the context based user-role Web service in our framework.

This task of integration is indeed quite simple. Instead of serving the entire eInstrument (which is the entire virtual representation of this instrument), we can use the context Web service from earlier discussions (see 4) to customize the Graphic User Interface sent to the user. This way we would achieve the

construction of the einstrument depending on the context, and thus dynamically as illustrated here 6.

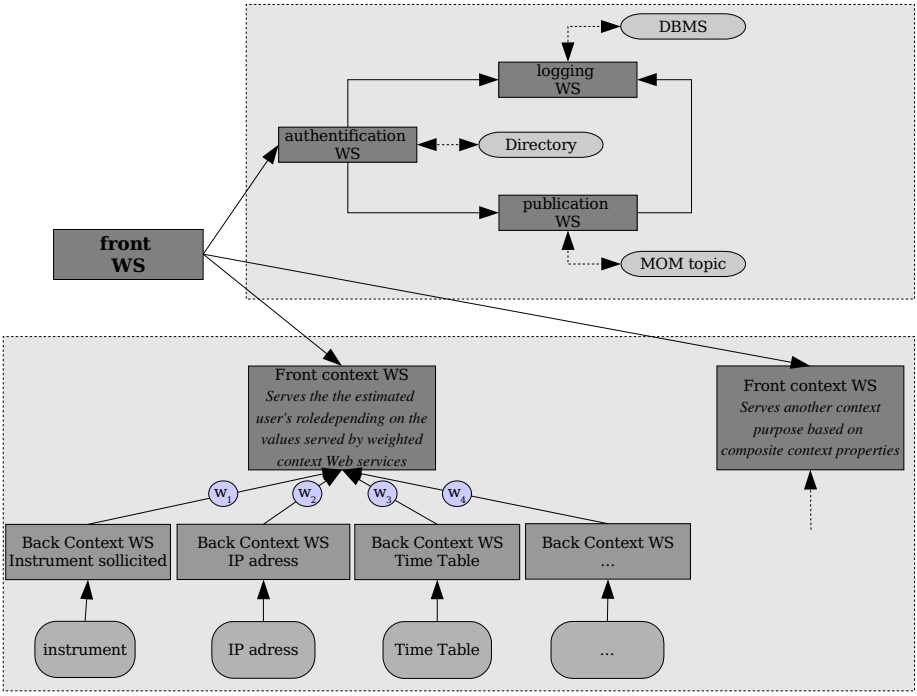

Fig. 6. A service-base context layer, ponderating context properties

5 Illustrations and examples about instruments being remotely controlled.

Just for illustration purpose, we will propose here a list of instruments that our laboratory is owning. The list of instrument concerned by the project of eInstrumentation is

- a *Network analyser*
- a *Hyperfrequency analyser*
- a *Optic fiber stretcher*
- a *Hyperfrequency ellipsometer*

The reader *must* understand that those instruments are in the same field of interest which is Hyperfrequency measure in a short description.
Nevertheless, it has to be considered that the sharing is operated not at in-

strument level, but at the application accessing the instrument. Virtually, this means that one can use our framework, not only to access an applications controlling an instrument, but to any C/C++ dll or Java based applications that could possibly do anything, even which will not access any instrument but address any other kind of resource.

Summary 3 *To sum up, depending on some context properties such as (not an exhausted list)*

- *the IP address the user's application is using*
- *the instrument that is being accessed*
- *the time when the user is connecting to the eInstrument*

The architecture can so determine the role the user is supposed to play in the session initiated. That is the reason why the displayed graphic controlls can be automatically *adapted to the user depending on the* context *of the session he is building. Nevertheless, there is no denying that one can hardly be convinced, that it is completely sure to correctly guess, at every connections and for all users, the corresponding role to support, relying on a set of few context properties. That is because none may be completely revealing for every situation (*i.e. every context !*). However, it has to be noticed that some context properties could be more revealing than others and so be more ponderated for role-evaluation time. Of course, because the evaluation could result in a bad interpretation of the context, there must be a* safety mechanism *offering the possibility to the user to switch, in our example, from the teaching interface to the research one. (that should be called "role switching" regarding the context). Moreover, this context Web service, for the role of the user, can be extended to other* context services. *In our framework, the context Web services are integrated by adding a new control layer at the initiation of the connection as shown in figure 6, representing our entire framework.*

6 Conclusion.

This paper intended to show a complete use case of Web services serving an entire framework as an interface for publishing messages *and* as an interface for adapting the service depending on the context (in fact, this paper focused on the evaluation of the role played by a user who establishes a session.
During this establishment period, one has to take into account the context in which the user is accessing the framework, this to provide an adapted service. In order to respect this requirement, the architecture go through different steps:

- only *one Web service* making up the publication of messages through a Message Oriented Middleware
- a *composite service* meaning the service is supplied by different Web service put altogether
- a *context layer*, at which a "context web service" take decisions regarding the context. This context Web service uses a composite service, built with single Web services, each accessing a context property.

Nevertheless, questions can be raised and may be matter of future works. For example, one can argue that establishing the ponderation to each Web service accessing a context property is not easy and there may be an best set of weights that optimize the success of the decisions of the context Web services. This means that there may be a way to minimize an erroneous decision based on the context properties.

In addition, is there a point where the changes in context affect the ponderation ? This means, is there any best set of ponderation for a given context ? This would tend to affirm that it would be possible to set up a "meta-context" that will, regarding the context itself, set weights on context properties differently.

Finally, our article discuss the evaluation of the role of the user at session establishment. Two things can be enlighten: from one hand one can pretend to integrate in our framework other context Web services determining another property in the system, in the other hand one can periodically re-evaluate the context so that this evaluation will not only be restricted at one determined time, but will periodically be called into question during the entire session time. This could be achieve by taking into account the click speed of the user, for example.

References

[1] Bernstein, Philip A. *Middleware: A Model for Distributed System Services*, Communications of the ACM, February 1996, Vol. 39, No.2, 86-98.

[2] Rachid Guerraoui, Andre Schiper, *Consensus: The Big Misunderstanding*, 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97), 1997, 183.

[3] Sangmi Lee, Sunghoon Ko, Geoffrey Fox, Kangseok Kim and Sangyoon, *A Web Service Approach to Universal Accessibility in Collaboration Services*, in Proceedings of 1st International Conference on Web Services, Las Vegas, (ICWS '03), June 2003

[4] Stephan Lukosch, Jrg Roth, *Reusing Single-user Applications to Create Multi-user Internet Applications*, Innovative Internet Computing Systems (I2CS), 2001, 79-90

[5] F. Pianegiani, D. Macii and P. Carbone, *Management of Distributed Measurement Systems Based on Abstract Client-Server Paradigms*, University of Trento Technical Report, 2004

[6] J. Rykowski and W. Cellary, Virtual Web Services - *Application of Software Agents to Personalization of Web Services*, Sixth International Conference on Electronic Commerce (ICEC'04), 2004, 409-418

[7] "SOAP", http://www.w3.org/TR/soap/

[8] "JMS: Java Messaging Service", Sun Microsystems, http://java.sun.com/products/jms/

[9] "JORAM", http://joram.objectweb.org/